# Can Large Language Models Write Parallel Code?

Daniel Nichols, Joshua H. Davis, Zhaojun Xie, Arjun Rajaram, Abhinav Bhatele
University of Maryland

PARALLEL SOFTWARE AND SYSTEMS GROUP

UNIVERSITY OF MARYLAND

# Timeline and Motivation

I wonder if these are any good at HPC?

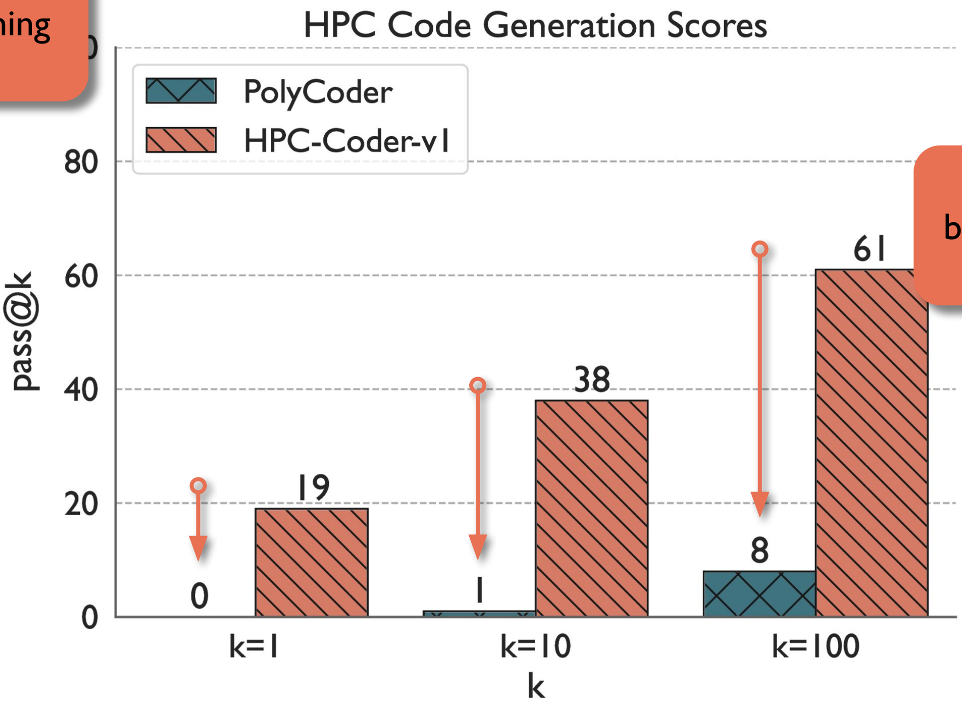
**August 2021**
OpenAI Codex Release

**November 2022**
ChatGPT Release

**January 2023**
We train HPC-Coder [ISC24]

# HPC-Coder: Improving Code LLMs for HPC

Much better code generation after fine-tuning on HPC data…




but these were simple tasks.

# Takeaways from HPC-Coder

LLMs are bad at Parallel Cod

We prompted the LLM to write simple parallel code…

```
/*
    Use MPI to compute the average of X across ranks
    and return the result on all ranks.
    Use the default communicator.
*/
double mpiAverage(double X) {
```

and usually got incorrect output.

```
    int rank;
    int nprocs;
    double average;
    MPI_Status status;
    MPI_Request request;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &nprocs);

    if (rank == 0) {
        MPI_Recv(&status, 1, MPI_INT, 0, 0,
            MPI_COMM_WORLD,
                 &request);
        MPI_Send(&X, 1, MPI_INT, 0, 0,
            MPI_COMM_WORLD);
        MPI_Send(&average, 1, MPI_DOUBLE, 0, 0,
                 MPI_COMM_WORLD);
    }
    return average;
}
```

We observed an ability to generate parallel code constructs, but not use them correctly

# Timeline and Motivation

**August 2021**
OpenAI Codex Release

I wonder if these are any good at HPC?

**November 2022**
ChatGPT Release

**January 2023**
We train HPC-Coder [ISC24]

They're not, but we can improve them

**March 2023**
GPT-4 Release

SotA Code LLM, we re-evaluate

**August 2023 - Now**
StarCoder, DeepSeek, CodeLlama, …

# The Need for New Benchmarks and Metrics

- LLMs are bad with parallel code
  - But how bad?
- State-of-the-art changes too quickly
  - We need a way to understand the current state of the practice
- HPC has unique software engineering objectives
  - Current benchmarks are mostly simple Python, Java functions
  - Performance, parallelism, efficiency are not the focus of current code LLM research

# Research Questions

**RQ1** – *How well do state-of-the-art LLMs generate parallel code, and which models are the best?*

**RQ2** – *Which parallel execution models and problem types are most challenging for LLMs?*

**RQ3** – *How performant and scalable is the parallel code generated by LLMs?*

**RQ4** – *How well can LLMs translate between execution models? How performant and scalable is the translated code?*

# What are code LLM benchmarks?

```
/* Return the value of the smallest odd
   number in the vector x.
   Use Kokkos to compute in parallel.
   Assume Kokkos is already initialized.
   Examples:

   input: [7, 9, 5, 2, 8, 16, 4, 1]
   output: 1
   input: [8, 36, 7, 2, 11]
   output: 7
*/
int smallestOdd(Kokkos::View<const int*> const& x) {
```

The LLM is given a natural language description of the problem…

# What are code LLM benchmarks?

```
/* Return the value of the smallest odd
   number in the vector x.
   Use Kokkos to compute in parallel.
   Assume Kokkos is already initialized.
   Examples:

   input: [7, 9, 5, 2, 8, 16, 4, 1]
   output: 1
   input: [8, 36, 7, 2, 11]
   output: 7
*/
int smallestOdd(Kokkos::View<const int*> const& x) {
   int min_odd = std::numeric_limits<int>::max();
   Kokkos::parallel_reduce("smallestOdd", x.size(),
      KOKKOS_LAMBDA(const int i, int& update_min_odd) {
         if (x(i) % 2 != 0 && x(i) < update_min_odd) {
            update_min_odd = x(i);
         }
      },
      Kokkos::Min<int>(min_odd));
   return min_odd == std::numeric_limits<int>::max()? -1 : min_odd;
}
```

The LLM is given a natural language description of the problem…

and generates the code to solve that problem.

Unit tests evaluate the correctness of the generated code.

# ParEval: A Parallel Code Generation Evaluation Framework

- 420 parallel code problems
  - 7 execution models
    - Serial, OpenMP, MPI, MPI+OpenMP, CUDA, HIP, Kokkos
  - 12 computational problem types
    - Sort, scan, dense linear algebra, sparse linear algebra, search, reduce, histogram, stencil, graph, geometry, fourier transform, transform
  - 5 problems per problem-type–execution model pair
- Drivers to evaluate correctness, performance, and scaling

# Another Example

```
/* Return true if `val` is only in one of vectors x or y.
   Return false if it is in both or neither. Use MPI to search in parallel.
   Assume MPI has already been initialized.
   Every rank has a complete copy of x and y.
   Return the result on rank 0.
   Examples:

      input: x=[1,8,4,3,2], y=[3,4,4,1,1,7], val=7
      output: true

      input: x=[1,8,4,3,2], y=[3,4,4,1,1,7], val=1
      output: false
*/
bool xorContains(std::vector<int> const& x, std::vector<int> const& y, int val) {
```

problem description

parallel instructions

examples

function header

# How do we evaluate LLM generated code?

- pass@k – How often are outputs correct?

Number of samples generated per prompt

$$\left[1 - \binom{N - c_p}{k} \Big/ \binom{N}{k}\right]$$

Number of correct samples for prompt $p$

**prompt**

```
/* Compute the dot product of x and y
using OpenMP */
int product(int *x, int *y, size_t N) {
```

**LLM**

**response 1**

```
int sum = 0;
#pragma omp parallel for
for (int i = 0; i < N; i++) {
   sum += x[i] * y[i];
}
return sum;
```

**response 2**

```
int sum = 0;
for (int i = 0; i < N; i++) {
   sum += x[i] * y[i];
}
return sum;
```

⋮

**response k**

What is the probability at least one of k responses is correct?

# How do we evaluate LLM generated code?

- speedup@k – What's the expected max speedup?

runtime of sequential baseline for prompt $p$

$$\mathbb{E}\left[\max\left\{\frac{T_p^*}{T_{p,s_1,n}},\ldots,\frac{T_p^*}{T_{p,s_k,n}}\right\}\right]$$

runtime of samples $s_1$ thru $s_k$ of prompt $p$ on $n$ resources

**prompt**

```
/* Compute the dot product of x and y
using OpenMP */
int product(int *x, int *y, size_t N) {
```

**LLM**

**response 1**

```
int sum = 0;
#pragma omp parallel for
for (int i = 0; i < N; i++) {
   sum += x[i] * y[i];
}
return sum;
```

**response 2**

```
int sum = 0;
for (int i = 0; i < N; i++) {
   sum += x[i] * y[i];
}
return sum;
```

⋮

**response k**

What is the *expected max speedup* over k responses?

# How do we evaluate LLM generated code?

- speedup@k – What's the expected max speedup?

$$\text{speedup}_n@k = \frac{1}{|P|}\sum_{p\in P}\sum_{j=1}^{N}\frac{\binom{j-1}{k-1}}{\binom{N}{k}}\frac{T_p^*}{T_{p,j,n}}$$

**prompt**

```
/* Compute the dot product of x and y
using OpenMP */
int product(int *x, int *y, size_t N) {
```

**LLM**

**response 1**

```
int sum = 0;
#pragma omp parallel for
for (int i = 0; i < N; i++) {
   sum += x[i] * y[i];
}
return sum;
```

**response 2**

```
int sum = 0;
for (int i = 0; i < N; i++) {
   sum += x[i] * y[i];
}
return sum;
```

⋮

**response k**

What is the *expected max speedup* over k responses?

# How do we evaluate LLM generated code?

- Correctness
  - pass@k
- Performance
  - $\text{speedup}_n$@k
  - $\text{speedup}_{max}$@
  - $\text{efficiency}_n$@k
  - $\text{efficiency}_{max}$@

What is the probability at least one of k responses is correct?

$$\text{pass}@k = \frac{1}{|P|}\sum_{p\in P}\left[1-\binom{N-c_p}{k}\Big/\binom{N}{k}\right]$$

What is the *expected max speedup* over k responses on *n* resources?

$$\text{speedup}_n@k = \frac{1}{|P|}\sum_{p\in P}\sum_{j=1}^{N}\frac{\binom{j-1}{k-1}}{\binom{N}{k}}\frac{T_p^*}{T_{p,j,n}}$$

What is the *expected max speedup* over k responses on *all* resource counts?

$$\text{speedup}_{\text{max}}@k = \frac{1}{|P|}\sum_{p\in P}\sum_{\substack{j=1\\ n\in\text{procs}}}^{N\cdot|\text{procs}|}\frac{\binom{j-1}{k-1}}{\binom{N\cdot|\text{procs}|}{k}}\frac{T_p^*}{T_{p,j,n}}$$

What is the *expected max efficiency* over k responses on *n* resources?

$$\text{efficiency}_n@k = \frac{1}{|P|}\sum_{p\in P}\sum_{j=1}^{N}\frac{\binom{j-1}{k-1}}{\binom{N}{k}}\frac{T_p^*}{n\cdot T_{p,j,n}}$$

What is the *expected max efficiency* over k responses on *all* resource counts?

$$\text{efficiency}_{\text{max}}@k = \frac{1}{|P|}\sum_{p\in P}\sum_{\substack{j=1\\ n\in\text{procs}}}^{N\cdot|\text{procs}|}\frac{\binom{j-1}{k-1}}{\binom{N\cdot|\text{procs}|}{k}}\frac{T_p^*}{n\cdot T_{p,j,n}}$$

# Choosing LLMs to Compare

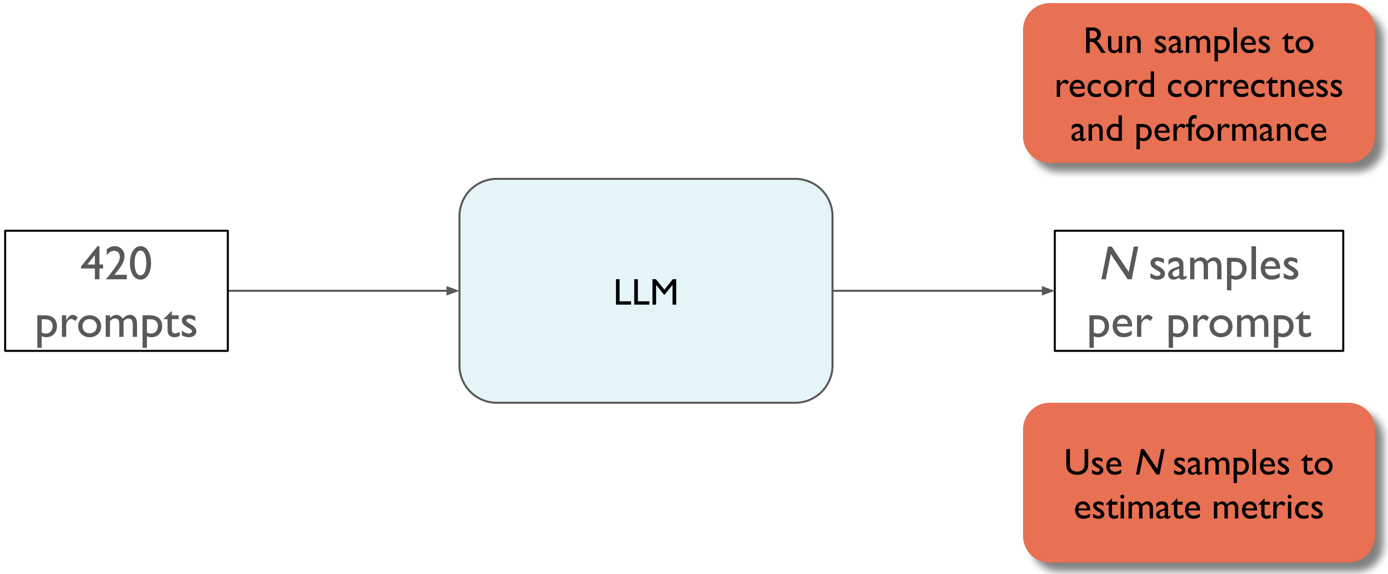
- SotA code LLMs
- Open and closed source, big and small

Two popular, standard benchmarks for Python code synthesis

| Model Name | No. of Parameters | Open-source Weights | License | HumanEval† (pass@1) | MBPP‡ (pass@1) |
|---|---|---|---|---|---|
| CodeLlama-7B [41] | 6.7B | ✓ | llama2 | 29.98 | 41.4 |
| CodeLlama-13B [41] | 13.0B | ✓ | llama2 | 35.07 | 47.0 |
| StarCoderBase [29] | 15.5B | ✓ | BigCode OpenRAIL-M | 30.35 | 49.0 |
| CodeLlama-34B [41] | 32.5B | ✓ | llama2 | 45.11 | 55.0 |
| Phind-CodeLlama-V2 [39] | 32.5B | ✓ | llama2 | 71.95 | — |
| GPT-3.5 [8] | — | ✗ | — | 61.50 | 52.2 |
| GPT-4 [34] | — | ✗ | — | 84.10 | — |

Best model at time of writing

# Using ParEval to Evaluate an LLM

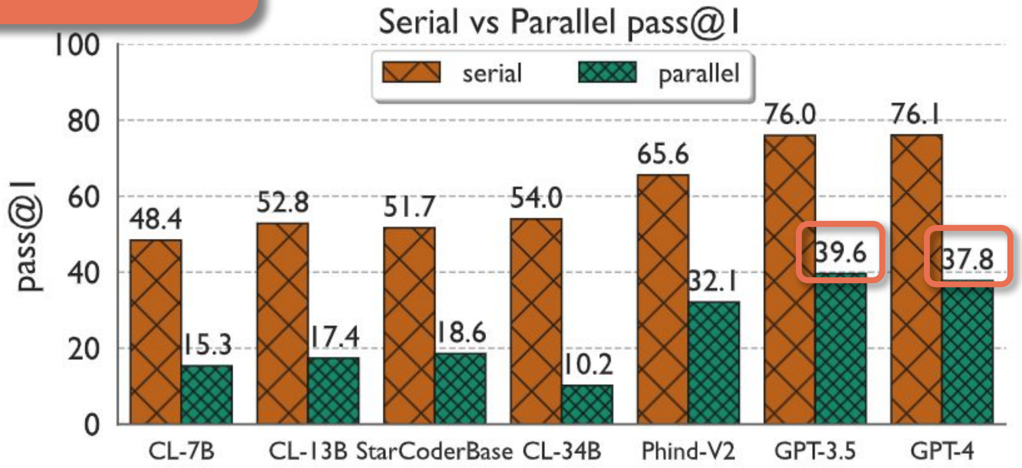


# ParEval Code Generation Results: Correctness

**RQ1**

*How well do state-of-the-art LLMs generate parallel code, and which models are the best?*

All of the models are bad at writing parallel code.




Unfortunately commercial models are the best.

PSSG

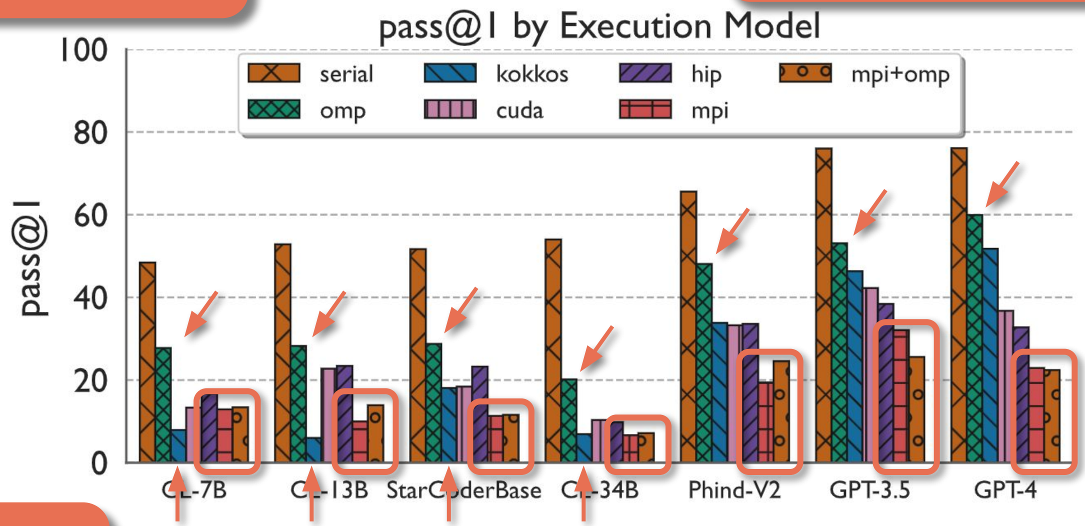
UNIVERSITY OF MARYLAND

# ParEval Code Generation Results: Correctness

LLMs are better for execution models "closer" to serial code.

They're bad with distributed memory.

**RQ2**
*Which parallel execution models and problem types are most challenging for LLMs?*




Small LLMs struggle with "low-data" execution models.

PSSG

UNIVERSITY OF MARYLAND

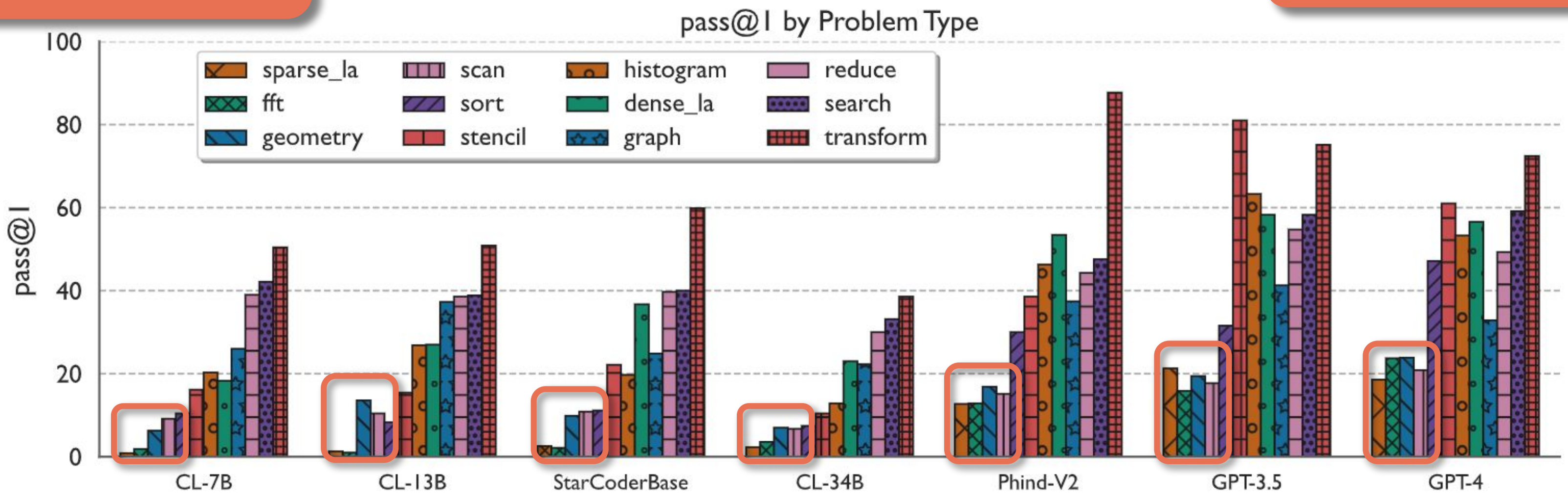
# ParEval Code Generation Results: Correctness

**RQ2**

*Which parallel execution models and problem types are most challenging for LLMs?*

LLMs struggle with sparse, unstructured problems.

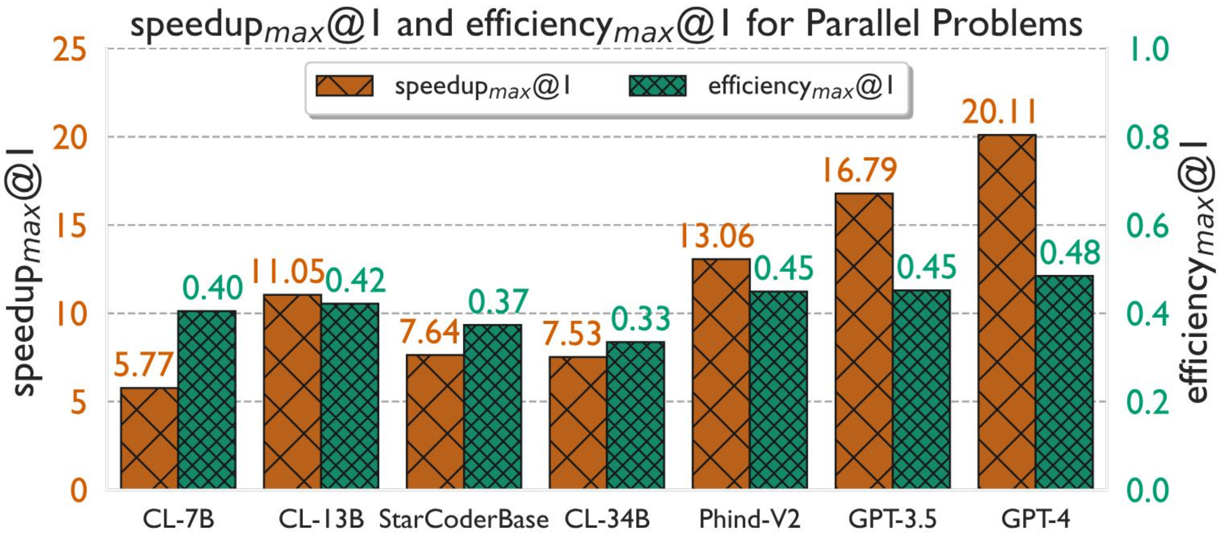
Surprisingly bad with sort and scan.



# ParEval Code Generation Results: Performance

Some LLMs produce much faster code than others even when less accurate.

**RQ3**
*How performant and scalable is the parallel code generated by LLMs?*



# Testing LLM Translation

- How well can LLMs translate between execution models?
- Translation tasks
  - Serial to OpenMP
  - Serial to MPI
  - Cuda to Kokkos

# Testing LLM Translation

```cpp
// A serial implementation of sumOfMinimumElements
/* Return the sum of the minimum value at each index
   of vectors x and y for all indices.
   i.e. sum = min(x_0, y_0) + min(x_1, y_1)+ ...
   Example:

   input: x=[3, 4, 0, 2, 3], y=[2, 5, 3, 1, 7]
   output: 10
*/
double sumOfMinimumElements(std::vector<double> const& x,
std::vector<double> const& y) {
   double sum = 0.0;
   for (size_t i = 0; i < x.size(); ++i) {
      sum += std::min(x[i], y[i]);
   }
   return sum;
}
```

```cpp
// An OpenMP implementation of sumOfMinimumElements
/* Return the sum of the minimum value at each index
   of vectors x and y for all indices.
   i.e. sum = min(x_0, y_0) + min(x_1, y_1) + ...
   Use OpenMP to sum in parallel.
   Example:

   input: x=[3, 4, 0, 2, 3], y=[2, 5, 3, 1, 7]
   output: 10
*/
double sumOfMinimumElements(std::vector<double> const& x,
std::vector<double> const& y) {
```

PSSG

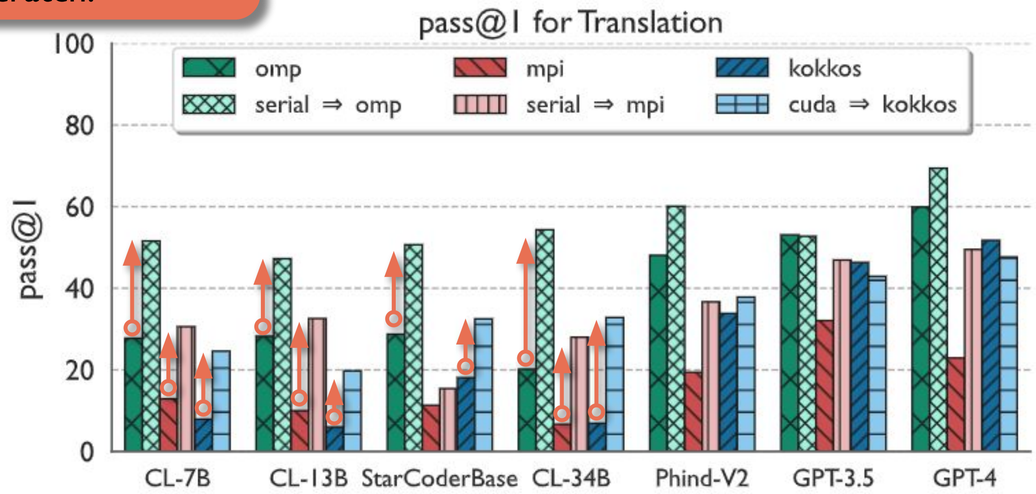
UNIVERSITY OF MARYLAND

# Translation Results

Almost always better at translating than generating from scratch.

**RQ4**

*How well can LLMs translate between execution models? How performant and scalable is the translated code?*




Small LLMs benefit significantly from serial examples.

# Contributions

- ParEval: a benchmark for comprehensively evaluating the ability of LLMs to generate parallel code
- Novel metrics for evaluating the performance of LLM generated code
- An in-depth study of SotA LLMs across ParEval and an identification of areas where improvement and future work is needed

**Can Large Language Models Write Parallel Code?**

**sometimes…**

# The Future of ParEval

dnicho@umd.edu

- Adding new tests
  - Fill-in-the-middle
  - Raja, Python (mpi4py, multiprocessing)
  - We are welcome to suggestions and contributions
  - https://github.com/parallelcodefoundry/ParEval/
- Up-to-date dashboard
  - https://pssg.cs.umd.edu/blog/2024/pareval/

ParEval

Dashboard

RLPF

"Performance-Aligned LLMs for Generating Fast Code" arXiv 2404.18864

PSSG

UNIVERSITY OF MARYLAND